Development of Computer Graphics Learning Tool Object-Based: A Case Study on Bresenham and Midpoint Algorithms

Maria Ulfah S **Ahmad Athaullah** **M. Rifqi Maarif**

Informatic's Department, Faculty of Science and Technology
State Islamic University Yogyakarta

Abstract – *This paper presents the development of computer graphics learning tool object-based, which in this paper is to visualize Bresenham and Midpoint Algorithms. The development first introduces the implementation of problems by using UML notation. The description of the UML consists of the use of Activity Diagram, Sequence Diagram, Use-Case Diagram, Class Diagram, and State Diagram. Then, implementation continues, which is to code a program. The objective of developing this Computer Graphics learning tool is to ease students for understanding Bresenham and Midpoint Algorithms.*

Keywords: Computer Graphics, Bresenham, Midpoint, UML.

1 Introduction

Computer Graphics is a part of computer science that relate with drawing and manipulating image digitally [1]. This science is composed by some parts; one of them is Geometry, which is a science for drawing a surface area.

Computer Graphics has been applied in many aspects of human life. One of them is in education and training [2]. In spite of that, notably for learning in a class, many algorithms are still presented with text and picture forms that cannot be simulated directly by using computer.

Based on those problems, students often feel such a hard for deep understanding on functions of those algorithms. This is because computer graphics algorithms especially Geometry are presented in forms of mathematics formula that abstract.

Therefore, this paper presents the development a helping tool for visualizing some of computer graphics algorithms, in particular are Bresenham and Midpoint Algorithms. The objective of this development is to help students for easier understanding of those algorithms so that can apply them to future development.

In this paper, those algorithms are chosen because both of them can be used to draw either line or circle where they are one of primitives on computer graphics [3]. Furthermore, the task of drawing straight line on a graphics screen is a fundamental building block for most of computer graphics applications [4].

2 Background Theories

Following will be presented Bresenham and Midpoint Algorithm and lastly is UML. First is Bresenham Algorithm.

2.1 Bresenham

Bresenham is an efficient algorithm to render a line with pixels [5]. The long dimension is incremented pixel by pixel, and the fractional slope is accumulated [6]. This algorithm is an algorithm to determine where points in an n-dimensional raster should be plotted in order to form a close approximation to a straight line between two given points [7].

The algorithm is commonly used to draw lines on a computer screen. Because of using only integer addition, subtraction and bit shifting, the operation of that algorithm is very cheap. Moreover, it does not need to round off the value of pixel position every time [8], so it can speed up the operation time. A minor extension to the original algorithm also deals with drawing circles [9].

2.1.1 Bresenham Algorithm [10] & [11]

Here is the basic Bresenham Algorithm. If we want to draw a line on a raster grid where we restrict the allowable slopes of the line to the range $0 \le m \le 1$. And next, if we restrict the line-drawing routine so that it always increments x as it plots, it becomes clear that, having plotted a point at (x, y), the routine has a severely limited range of options as to where it may put the next point on the line:

- It may plot the point $(x+1, y)$, or:
- It may plot the point $(x+1, y+1)$.

Therefore, by working in the first positive octant of the plane, line drawing becomes a matter of deciding possible point between two possibilities at each step. We can draw a diagram of the situation, which the plotting program finds itself in having plotted *(x, y)*.

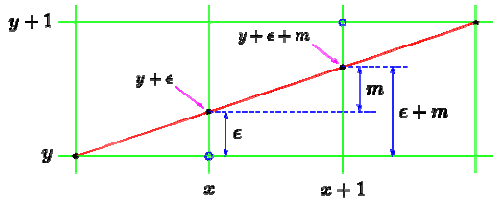

Figure 1. Bresenham line

The plotted point (*x*, *y*) usually will be in error. Actually, mathematical point on the line will not be addressable on the pixel grid. Therefore, we associate an error, ε, with each *y* ordinate; the real value of *y* should be *y* + ε. This error will range from -0.5 to just under +0.5.

In moving from *x* to *x*+1, the value of the true *y* will be increased by an amount equal to the slope of the line, *m*. If we choose to plot (*x*+1, *y*) and if the difference between this new value and *y* is less than 0.5, then the formula is:

$$\mathbf{y + \varepsilon + m < y + 0.5} \qquad (1)$$

Meanwhile, if we choose to plot (*x*+1, *y*+1), the total error will be minimize between line segment. This resulting error can now be written back into ε. Therefore, it will allow us to repeat the whole process for the next point along the line, at *x*+2.

The new value of error can adopt one of two possible values, depending on what new point is plotted. If (*x*+1, *y*) is chosen, the new value of error is given by:

$$\varepsilon_{new} \leftarrow \mathbf{(y + \varepsilon + m) - y} \qquad (2)$$

If (*x*+1, *y*+1) is chosen:

$$\varepsilon_{new} \leftarrow \mathbf{(y + \varepsilon + m) - (y + 1)} \qquad (3)$$

In algorithm notation, the processes are:

1. Define two points that will be connected to draw a line
2. Define one of those points, which in left position, as origin point, (x_0, y_0) and another as end point (x_1, y_1)
3. Compute d*x*, d*y*, 2d*x* and 2d*y* – 2d*x*
4. Compute parameter: $p_0 = 2dy - dx$
5. For every x_k along the line, starts with $k = 0$,
   - if pk < 0, then the next point is (x_k+1, yk), and $p_{k+1} = p_k + 2dy$
   - otherwise, then the next point is $(x_k + 1, y_k + 1)$, and $p_{k+1} = p_k + 2dy – 2dx$
6. Repeat from number 5 to define next pixel position, until $x = x_1$ and $y = y_1$.

2.2 Midpoint

Midpoint Circle Algorithm is also called Bresenham Circle Algorithm [12]. The Midpoint is the middle point of a line segment [13]. Therefore, it is equidistant from both endpoints. Its formula in the plane segment, with endpoints (x_1, y_1) and (x_2, y_2) is:

$$\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \qquad (3)$$

For an n-dimensional space with axes x_1, x_2, x_3… x_n, the midpoint of an interval is given by:

$$\frac{x_{11} + x_{12}}{2}, \frac{x_{21} + x_{22}}{2}, \frac{x_{31} + x_{32}}{2}, \ldots, \frac{x_{n1} + x_{n2}}{2} \qquad (4)$$

Computation to draw circle is started by identifying parts of circle by using symmetry property. This is done by dividing circle into parts with 45° angle. Therefore, there are eight parts.

The same as Midpoint Algorithm for drawing line, then in this phase, there are two pixels that must be chosen. They are (*x*+1, *y*) or (*x*+1, *y*-1).

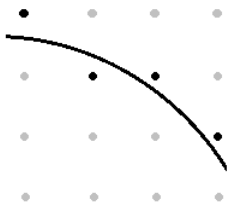

Figure 2. Circle using midpoint algorithm

Next step is to define circle equation and function to set determine variable. The formula for circle with centre (0, 0) is :

$$f(x, y) = x * x + y + y - r * r = 0 \qquad (5)$$

The function f(*x*, *y*) of above formula will have positive value if point (*x*, *y*) is outside from circle, and otherwise if point (*x*, *y*) is inside from circle. Whereas determine variable function and increment is given by these formulas:

$$g(x, y) = (x + 1) * (x + 1) + (y - ½) * (y - ½) - r * r \qquad (6)$$

$$\text{DeltaG1} = 2x + 3 \qquad (7)$$

$$\text{DeltaG2} = 2x - 2y + 5 \qquad (8)$$

Differ with increment value of straight-line algorithm which constant; on circle curve, it is not constant. Commonly, polynomial curve order *n* needs *n* level increment. At original point (x_1, y_1), determine variable computation has real parts, so that integer number

operation cannot be used directly. In practice, this is finished by add ¼ value to determine variable. This will not interfere changes of number sign, as it operation is done by integer, and it needs faster operation.

2.3 UML

Unified Modeling Language (UML) is a standardized general purpose modeling language in the field of software engineering [14]. UML includes a set of graphical notation techniques to create abstract models of specific systems.

UML offers a standard way to write a system's blueprints, including conceptual components such as actors, business processes, system's components, and activities. As well as concrete things such as programming language statements, database schemas, and reusable software components.

UML help us to develop model from varies forms or types of software that is running on different operating system and network. Besides that, programmer can understand clearly purposes and aims of system design so software can be developed by using any programming language.

However, because UML is a model of software system development based on object so it uses class and operation form on basic concept. Therefore, it is suitable in programming to use language of object oriented such as C, C++, Java, VB and many others.

A model of software designed with UML is developed by using diagram and symbol forms to represent elements of system. Diagrams that are used are:

- *Use-case Diagram: shows the functionality in terms of actors, their goals represented as use cases, and any dependencies among those use cases.*
- *Class Diagram: describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.*
- *State Diagram: is a standardized notation to describe many systems, from computer programs to business processes.*
- *Sequence diagram: shows how Objects communicates with each other in terms of a sequence of messages. Also indicates the lifespan of objects relative to those messages.*
- *Collaboration Diagram: are types of activity diagram in which the nodes represent interaction diagrams.*
- *Activity Diagram: represents the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.*
- *Component Diagram: depicts how a software system is split up into components and shows the dependencies among these components.*
- *Deployment Diagram: serves to model the hardware used in system implementations, and the execution environments and artifacts deployed on the hardware.*

Object in software analysis and design is a concept, thing, and something to distinguish them from their environment. As simply, object like as a car, a human, an alarm and something else.

Nevertheless, object can be something abstract that is live in such as table, database, event, or system messages. The object is introduced by its state or operation. As example for a car, it can be recognized by its color, form, like wise human by its voice. These features will distinguish an object with other objects.

The reasons whether the nowadays development of software based on object, first is its scalability, where object can be used to draw easier a big and complex system. Second is dynamic modeling which can be used to model a dynamic and real time system.

UML is a model language that must be used simultaneously with methodology of software development [15]. This methodology is a guide for developing software. Moreover, this system will present Unified Software Development Process (USDP). USDP is an industry standard software development process that is free and generic process for the UML [16].

Iterations are the key of USDP. Each of iteration contains workflows. They are:

- Requirements: Purpose of this workflow is to collect data from clients. UML diagrams that are used are Activity Diagram and Use-Case Diagram.
- Analysis: In this workflow, requirements from clients will be defined. Sequence Diagram is used here.
- Design: Purpose of this work is to produce class design and its interaction that is class diagram.
- Implementation: Process that is happened here is to translate design into codes of program.
- Test: This workflow is used to evaluate system suitability by analysis, design or user requirement.

3 Methodology

The development methodology begins with requirement workflow to collect data from client. In the requirement workflow, activities that are happened are illustrated well in Figure 3. The activity diagram is created to represent the process to view the line or circle, by first input the points or radial. If a user does not key any of points or radial, no line or circle is resulted.

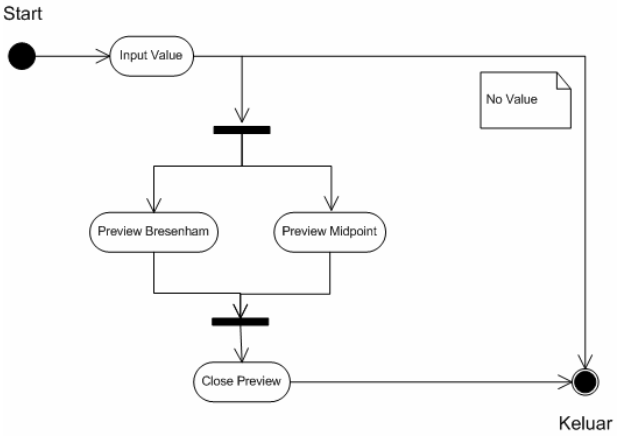

Figure 3. Activity diagram of system

Based on diagram in Figure 3, actor of the system can be defined. Use-case diagram for that actor can be seen as in Figure 4. In this system, actor is a user that can be a student, lecture or somebody that one to view line or circle by using Bresenham or Midpoint Algorithm. This use-case defines relationship between system and actor. The relationship can be an input from actor to system or an output from system to actor [17].

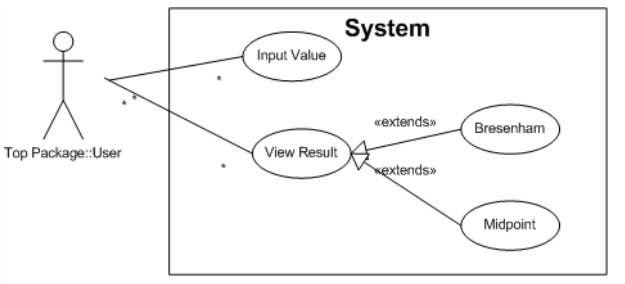

Figure 4. Use case diagram of user

The next process is analysis where data that are gathered from previous process will be analysis to bring to requirements of client. These requirements are pictured out on Sequence Diagram, as follow:

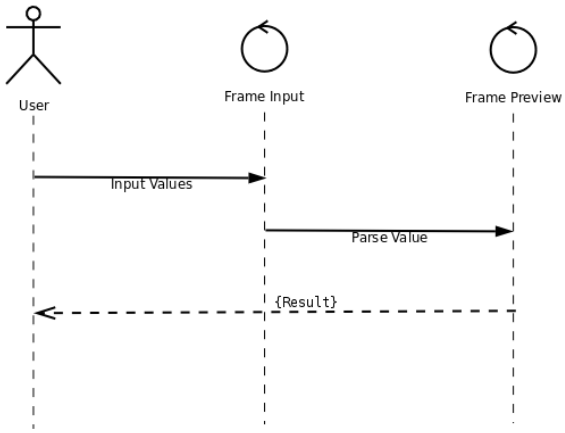

Figure 5. Sequence diagram of system

Class Diagram that is used in this system is Frame class and Main class. For more explanation, can be seen in Figure 6 and 7 respectively.

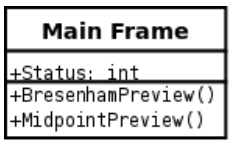

Figure 6. Class diagram: Frame

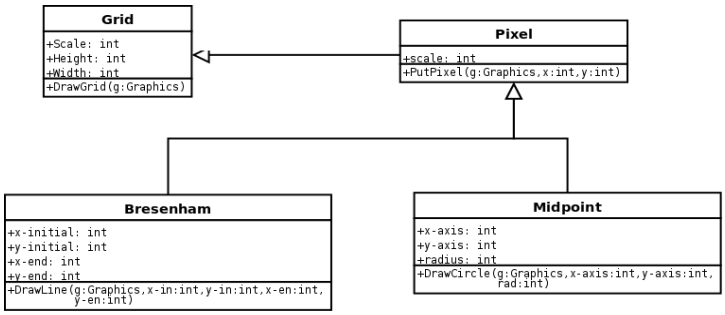

Figure 7. Class diagram: Main

State diagram is illustrated in Figure 8. This diagram consists of two transitions, whether user wants to view line by using Bresenham Algorithm and/or if user wants to view circle.

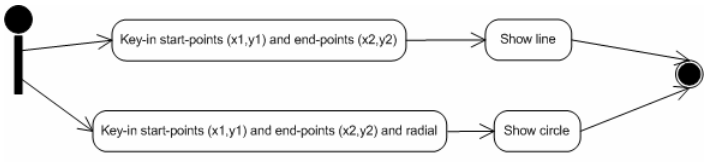

Figure 8. State diagram of system

In the implementation workflow, the interface is as simple as this figure. For both of algorithms, user is required to input points and especially for midpoint' circle is accompanied with radial.

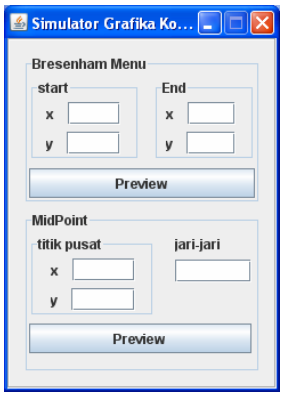

Figure 9. Interface of system

Some testing has been done to the system. Resulted line if user keys inputs as illustrated in Figure 10 is presented in Figure 11.

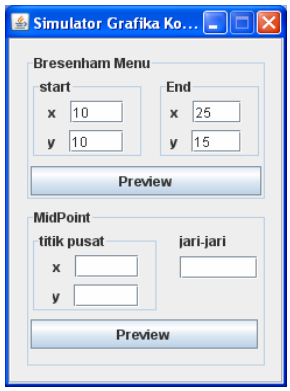

Figure 10. Input for drawing line

Resulted line is rendered in pixel grid. The line is showed not smooth. This is because size of the pixel is defined relatively big.

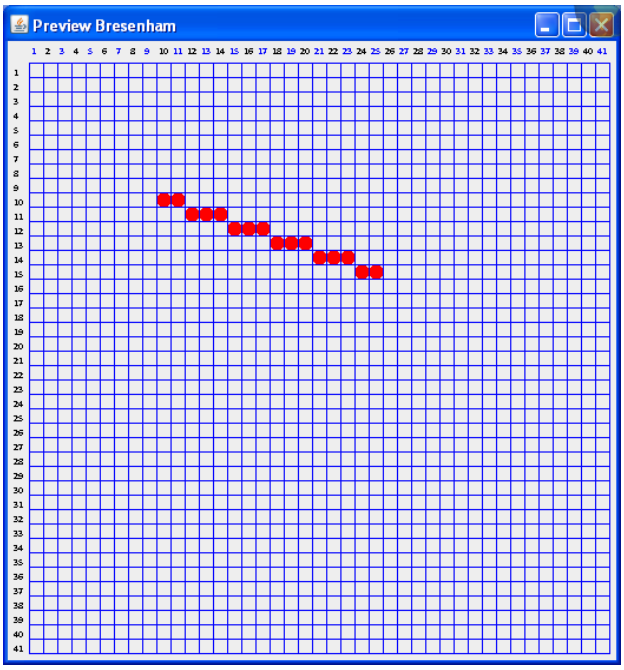

Figure 11. Resulted line

While, if user keys inputs for viewing circle by using Midpoint Algorithm, three inputs are keyed. Two of parameters are for *x* and *y* points of origin, and one parameter is for radial of circle, as illustrated in Figure 12.

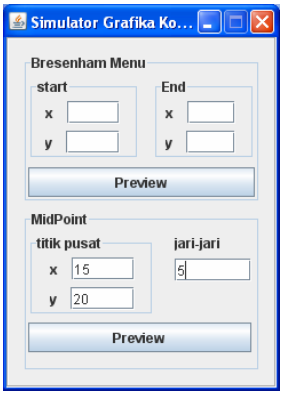

Figure 12. Input for drawing circle

The circle resulted can be seen in Figure 13. The smoothness of circle is not so good, the same with line drawing.

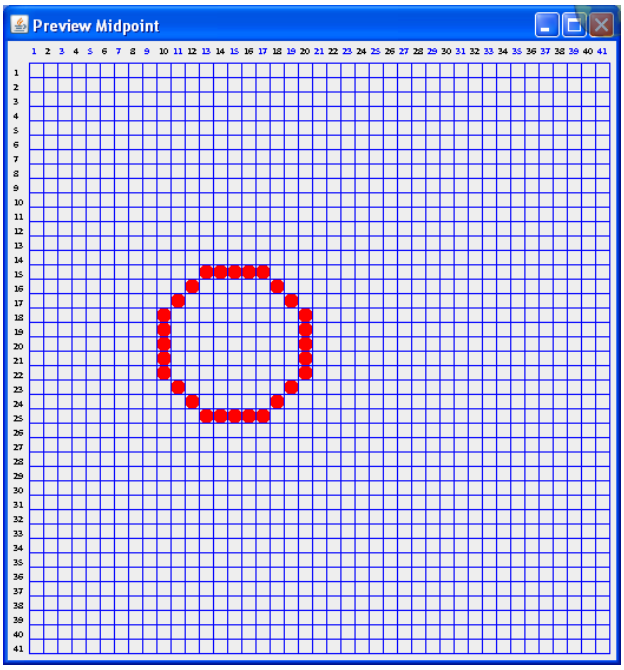

Figure 13. Resulted circle

4 Conclusions and Future Works

This paper has discussed the importance of developing a helping tool to ease students for understanding of some of computer graphics algorithms, which are Bresenham and Midpoint Algorithm. However, this system does not combine yet error checking for input.

Therefore, in future, it is hoped that more algorithms will be visualized and error checking will be implemented. Furthermore, the visualization will be presented nicely, such as use web-based, pixel grid that smooth, more number of pixel, interface that more user friendly, user guide, help menu, and many others.

References

[1] Anonim, Accessed from http://id.wikipedia.org/wiki/Grafika_komputer at 3rd of July 2009.

[2] Alfa Ryano, Accessed from http://elektroundip2002.files.wordpress.com/2008/02/tugas-01-grafkom.pdf at 3rd of July 2009.

[3] Djoni Haryadi Setiabudi, "Kurva bezier dan bresenham untuk pembuatan lingkaran", Jurnal Informatika, Vol. 2, No. 2, pp. 51-56, Nov. 2001.

[4] Koopman, P., "Bresenham line-drawing algorithm", Forth Dimensions, Vol. VIII, No. 6, pp. 12-16.

[5] Black, P.E., Accessed from http://www.itl.nist.gov/div897/sqg/dads/HTML/bresenham.html at 3rd of July 2009.

[6] Black, P.E., Accessed from http://www.itl.nist.gov/div897/sqg/dads/HTML/bresenham.html at 3rd of July 2009.

[7] Anonim, Accessed from http://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm at 3rd of July 2009.

[8] Ariesto Hadi Sutopo, *Pengantar grafika komputer*, Gava Media, Yogyakarta, pp. 49-51, 2002.

[9] Anonim, Accessed from http://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm at 3rd of July 2009.

[10] Flanagan, C., Accessed from http://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html at 3rd of July 2009.

[11] Ariesto Hadi Sutopo, *Pengantar grafika komputer*, Gava Media, Yogyakarta, pp. 51, 2002.

[12] Ariesto Hadi Sutopo, *Pengantar grafika komputer*, Gava Media, Yogyakarta, pp. 55, 2002.

[13] Anonim, Accessed from http://en.wikipedia.org/wiki/Midpoint at 3rd of July 2009.

[14] Anonim, Accessed from http://en.wikipedia.org/wiki/Unified_Modeling_Language at 3rd of July 2009.

[15] Shofwatul 'Uyun, et.al., "Sistem pemandu kenaikan pangkat dan jabatan dosen berbasis objek (Studi kasus di Fakultas Sains dan Teknologi UIN Yogykarta)", Compile: Jurnal Teknologi Komputer, Vol. 2, No. 1, pp. 71-83, Jan. 2009.

[16] Anonim, "3C05: Unified software development process", Zuelke Engineering, Accessed from http://www.zuelke.com at 3rd of July 2009.

[17] Fajar Saptono, "Pembuatan perangkat lunak SHORGA untuk menentukan jalur terpendek antar kota menggunakan algoritma genetika", Tugas Akhir UII, pp. 37, 2007.